2018 44th Euromicro Conference on Software Engineering and Advanced Applications

A Systematic Mapping Study on API Documentation Generation Approaches

Kristian Nybom, Adnan Ashraf, Ivan Porres
Åbo Akademi University
Faculty of Science and Engineering
Vesilinnantie 3, 20500 Turku, Finland
kristian.nybom@abo.fi, adnan.ashraf@abo.fi, ivan.porres@abo.fi

***Abstract*—Background: Application Programming Interfaces (APIs) are key to software reuse. Software developers can link functionality and behaviour found in other software with their own software by taking an API into use. However, figuring out how an API works is usually demanding, and may require that the developers spend a notable amount of time familiarizing themselves with the API. Good API documentation is of key importance to simplify this task.**
Objective: To present a comprehensive, unbiased overview of the state-of-the-art on tools and approaches for API documentation generation.
Method: A systematic mapping study on published tools and approaches that can be used for generating API documentation, or for assisting in the API documentation process.
Results: 36 studies on API documentation generation tools and approaches analyzed and categorized in a variety of ways. Among other things, the paper presents an overview of what kind of tools have been developed, what kind of documentation they generate, and what sources the documentation approaches require.
Conclusion: Out of the identified approaches, many contribute to API documentation in the areas of natural language documentation and code examples and templates. Many of the approaches contribute to ease API users' understanding and learning of the API, but also to the maintenance and generation of API documentation. Most of the approaches are automatic, simplifying the API documentation generation notably, under the assumption that relevant sources for the generation are available. Most of the API documentation approaches are evaluated either by exercise of the approach followed by analysis of the results, or by empirical evaluation methods.

I. Introduction

Software development is nowadays largely the process of integrating existing features and reusing them by writing client code interfacing to one or more Application Programming Interfaces (API) [1], [2]. Thus, APIs are the key to software reuse: they allow programmers beyond the original developers to use a certain component or service. Although the official API documentation often is a sufficient source of information when a developer takes a new API into use, online discussion forums – such as Stack Overflow – often provide more explanatory descriptions of specific API usages relevant to the developers, and are generally of good quality [3]. However, while the answers given at Stack Overflow may be syntactically correct, they are not without problems. In [4] a systematic analysis of the impact of information resources on the code security was carried out. The main findings reported were that developers relying only on Stack Overflow produced significantly less secure code than those relying on official API documentation only, while developers using API documentation produced significantly less functional code than those using Stack Overflow. While it is often notably faster and easier to find relevant information for a specific API from the Internet than from the official documentation, such crowd-sourced API knowledge is scattered around the Internet and disconnected from the official documentation [5].

Without proper tool support, creating and maintaining API documentation is a demanding task. Whenever the source code for the API is updated, the corresponding documentation needs to be updated as well. When code updates are done frequently, it is not uncommon that the necessary documentation updates are forgotten, or inadequate [6]. On the other hand, when developers take a new API into use for their software, finding the correct interface for a specific problem might be demanding because of the size of the documentation. Easily finding the correct resources in an API is therefore of importance.

All the things mentioned above point towards the same need: tools to create and maintain API documentation. With such tools, the expectation is that creating accurate API documentation is notably easier and faster, and this helps API users produce better software more quickly. This paper presents a systematic mapping study (SMS) on the research on approaches for API documentation generation. We begin by presenting how the SMS was designed in Section II. In Section III, we present the main findings of the SMS. In this Section we focus on answering the research questions, as defined in Section II. Section IV discusses the threats of validity of this study, and the paper is concluded in Section V.

II. The Systematic Mapping Study

This section presents the main points of the protocol created to conduct the SMS. The protocol itself is useful for replicability and validity review, as it gives a step-by-step description of how the study was performed. More specifically, it explains the goals for the research, the criteria for searching for original papers potentially relevant to the study, the criteria for including original papers in the study, how data was

978-1-5386-7383-6/18/$31.00 ©2018 IEEE
DOI 10.1109/SEAA.2018.00081

extracted from the original papers, and how the extracted data was synthesized.

In the remainder of this paper, we refer to *approaches* rather than to tools, since the word tool can be interpreted as a standalone application, but this paper is not limited to those. We also consider e.g. plug-ins to Software Development Kits (SDK's) and other forms of approaches that can assist in API documentation generation. However, we do not take into consideration API documentation *guidelines* and similar.

A. Research Questions

The research questions (RQ) are as follows:

RQ1: What approaches exist for creating new and improving existing API documentation?
RQ2: How do the approaches contribute to API documentation?
RQ3: What are the sources for API documentation?
RQ4: What are the quality properties of the approaches?
RQ5: How are the documentation approaches evaluated?

The first question (RQ1) is concerned with different ways of either creating new or improving existing API documentation. This RQ does not rule out approaches that may provide some kind of supplementary information regarding existing API documentation, which later on can be used for improving that documentation. RQ2 is aimed at finding out the expected results of the documentation approaches. In order to get a better understanding of the published approaches, RQ3 looks at the sources used by the approaches. We believe this is a relevant research question because it provides information about the sources that need to be available and preferably reliable, in order to obtain the expected results. RQ4 in turn aims at finding out why and how the identified approaches are useful in some way. The fifth and last RQ is concerned with whether and how the approaches have been evaluated, i.e., whether evidence is provided for improvements in API documentation. Figure 1 shows the API documentation generation process and how the research questions relate to it.

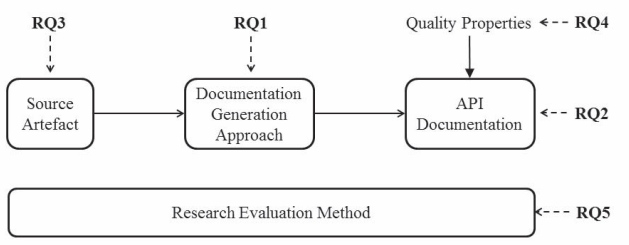

Fig. 1. Research questions and the documentation generation process.

Based on the RQs, the population, intervention, comparison, outcomes, and context (PICOC) is presented in Table I.

B. Search Strategy for Primary Studies

This section presents our search strategy. It is based on the Systematic Literature Review (SLR) guidelines in [7], [8].

TABLE I
PICOC

Aspect	Value
Population (P)	Software application developers
Intervention (I)	Approaches for creating useful API reference documentation
Comparison (C)	No comparison intervention
Outcomes (O)	An overview of approaches for creating and improving API reference documentation, and of their respective significance
Context (C)	Module, component, and service integration with client code

1) Search Terms: Table II lists the search terms used when searching for original papers for this study. The search terms are derived from the research questions and the PICOC in section II-A.

TABLE II
SEARCH TERMS WITH ALTERNATE SPELLINGS

Term	Alternate Spelling
API*	API, APIs
Application Programming Interface*	Application Programming Interface, Application Programming Interfaces
Librar*	Library, Libraries
Document*	Document, Documentation
Algorithm*	Algorithm, Algorithms
Approach*	Approach, Approaches
Method*	Method, Methods
Generat*	Generate, Generation
Autom*	Automatic, Automation, Automate
Evaluat*	Evaluate, Evaluation
Assess*	Assess, Assessment
Experiment*	Experiment, Experimental, Experimentation
Test *	Test, Testing
Empirical*	None

2) Search Strings: The search terms listed in Table II were combined into a search string for use in the digital libraries. The general search string was the following:

Search String
(API OR "Application Programming Interface*" OR Librar*) AND Document* AND (algorithm* OR approach* OR method* OR generat* OR creat* OR automat* OR evaluat* OR assess* OR study OR measur* OR experiment* OR test* OR empirical*)

3) Databases: The search string shown above was applied in the following digital libraries:

- IEEE Xplore
- ACM Digital library
- ScienceDirect
- SpringerLink

Since the digital libraries have different possibilities for defining search strings, it was customized to every digital library. From the collected results, duplicates were removed.

C. Study Inclusion Criteria

The inclusion criteria for primary studies were as follows:

- Written in English *AND*

- Published in a peer-reviewed journal, conference, or workshop of computer science, computer engineering, or software engineering *AND*
- Describing any of the following:
  - Methods or approaches for assisting in documentation search *OR*
  - Documentation approach or algorithm *OR*
  - Assessment or evaluation method or metrics for documentation

If several papers presented the same documentation approach, only the most recent was included, unless the contributions of those papers were different.

D. Title and Abstract Level Screening

In this phase, the inclusion criteria in Section II-C were applied to publication titles and abstracts. To minimize researcher bias, two researchers independently analyzed the search results. Afterwards, the results were compared and any disagreements were resolved through discussions. The filtered list of papers from this phase was used as input for the following phase. Due to the large number of papers found, we began by screening the titles and rejecting papers not fulfilling the inclusion criteria. After the title screening, we continued by screening the abstracts of the accepted publications. Based on [9], we believe that this approach does not compromise our results.

E. Full Text Level Screening

In this phase, the remaining papers were analyzed based on their full text. Again, to minimize bias, two researchers applied the inclusion criteria in Section II-C on the full text. The results were compared and disagreements were resolved through discussions. The researchers also documented a reason for each excluded study [10].

F. Study Quality Assessment Checklist and Procedure

The selected papers were assessed based on their quality. One researcher assessed the quality of the selected papers. Any papers not meeting the minimum quality requirements were excluded from the set of primary studies. The output from this phase was the final set of papers. Only one researcher performed the quality assessment, and consequently this is a threat to the validity.

Table III presents the checklist for study quality assessment. For each question in the checklist, a three-level, numeric scale was used [10]. The levels were: yes (2 points), partial (1 point), and no (0 point). Based on the checklist and the numeric scale, each study could score a maximum of 34 and a minimum of 0 points. We used the first quartile ($34/4 = 8.5$) as the cutoff point for the inclusion of studies. Therefore, if a study scored 8 points or less, it was excluded due to its lack of quality with respect to this study. The researcher documented the obtained score of each included/excluded study.

We point out that the quality we assessed was in terms of relevance for this study, i.e., a paper excluded in this phase could in fact be a very good paper but simply not relevant for this study, or not having enough contributions for this study, and was therefore excluded. We also point out that the quality assesment checklist in Table III was designed to find all studies that in one way or another would contribute to the research questions of this SMS. Consequently, we did not expect to find studies that would score full points in this phase. Rather, with this assessment we wanted to find those studies that had enough contributions to help address the research questions. Therefore, we used the first quartile as the cutoff point, essentially meaning that a study was included if it completely answered four of the questions, and partially answered one more question.

TABLE III
STUDY QUALITY ASSESSMENT CHECKLIST, PARTIALLY ADOPTED FROM [10]

#	Question
Theoretical contribution	
1	Is at least one of the research questions addressed?
2	Was the study designed to address some of the research questions?
3	Is a problem description for the research explicitly provided?
4	Is the problem description for the research supported by references to other work?
5	Are the contributions of the research clearly described?
6	Are the assumptions, if any, clearly stated?
7	Is there sufficient evidence to support the claims of the research?
Experimental evaluation	
8	Is the research design, or the way the research was organized, clearly described?
9	Is a prototype, simulation, or empirical study presented?
10	Is the experimental setup clearly described?
11	Are results from multiple different experiments included?
12	Are results from multiple runs of each experiment included?
13	Are the experimental results compared with other approaches?
14	Are negative results, if any, presented?
15	Is the statistical significance of the results assessed?
16	Are the limitations or threats to validity clearly stated?
17	Are the links between data, interpretation and conclusions clear?

G. Data Extraction Strategy

We used the form shown in Table IV to extract data from the primary studies. One researcher extracted the information from the papers, and the extracted data was then used for analysis. The data that was extracted was such data that it would answer the research questions in this paper.

H. Synthesis of the Extracted Data

The extracted data from the papers was used for analysis, in order to obtain a high-level view of different aspects related to the documentation approaches. The papers were categorized in different ways, and collective results were extracted. The results from this phase are presented in Section III.

III. RESULTS

In this section we present the main findings of this study. The search string used included words, such as ”Library” and ”Document”, which are used in many other contexts than API documentation. Also, the wording used in papers varies, e.g. some authors refer to software libraries, while others to APIs. Therefore, we used a search string (see Section II-B2) that

TABLE IV
DATA EXTRACTION FORM

Data item	Value	Notes
General		
Data extractor name		
Data extraction date		
Study identifier (S1, S2, S3, ...)		
Bibliographic reference (title, authors, year, journal/conference/workshop name)		
Author affiliations and countries		
Publication type (journal, conference, or workshop)		
API documentation related		
(RQ1) Type of approach (e.g. tool, plugin, web-based)		
(RQ1) Documentation generation method (automatic, sem-automatic, data mining, manual)		
(RQ2) Type of documentation generated (e.g. code examples, free text, formal specifications)		
(RQ3) Source for documentation generation (e.g. source code, user activity, free text, formal specifications)		
(RQ4) Attributes/usability of generated documentation (e.g. improved development time, reduced information finding time, improved understanding of complex API's)		
(RQ5) Evaluation method (analytical, empirical, simulation, execution of approach)		

would return all papers indifferently of the chosen wording. Consquently, the initial search produced a large number of papers as can be seen in Table V, which lists the number of papers to be processed in each phase. The majority of the papers found in the initial search were related to real physical libraries, and documentation in such libraries. Nevertheless, after the screening only 95 papers remained. At this point, based on the papers found we decided to slightly modify the set of research questions in order to keep the SMS more focused. In effect, we removed some research questions concerning API documentation usability, which we felt would be difficult to answer based on the set of primary studies. This is the main reason to the big difference in the number of papers left after the full text screening (95 papers) and the quality assessment (36 papers). Another reason is that we have not included studies dated before 2000 in this SMS. Had we focused on the current topic from the beginning, the total number of papers to process, specifically in the intermediate phases, would have been smaller.

TABLE V
NUMBER OF PAPERS IN EACH PHASE OF THE PAPER SEARCH AND SCREENING

Phase	Number of papers
Initial search results without duplicates	1899
After title and abstract screening	122
After full text screening	95
After quality assessment	36

Before moving on to the individual research questions, we note that the majority of the identified tools and approaches for supporting API documentation have been published as conference papers, while only a small number of them as journal papers. Figure 2 illustrates this distribution.

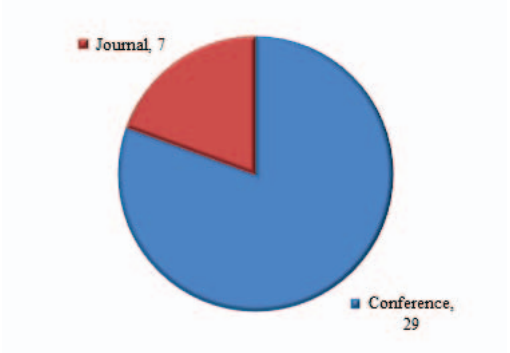

Fig. 2. Distribution and paper count of publication forums

A. Approaches for API documentation (RQ1)

Table VI lists the final set of primary studies included, with their corresponding references, the types of approaches, and their names if specified. Most approaches were designed as tools, but a few plugins were also reported. We point out that not all primary studies presented tools as such. Some of the studies described documentation approaches without explicitly mentioning concrete implementations of those approaches.

From Figure 3, we can see that this topic has been interesting for the scientific community ever since the beginning of the millennium. Since 2002, this area of research has slowly been gaining pace, although with quite notable variations from year to year in the number of publications. We note that the number of papers in 2016, as illustrated in Figure 3, may be too small, since the initial paper search was done in the beginning of December 2016. There is therefore a small probability that a few papers from 2016 were not included in this study.

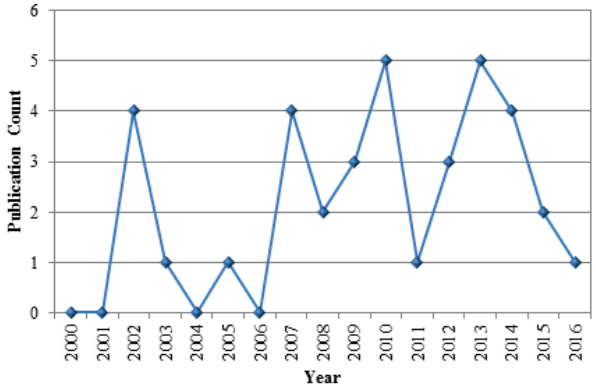

Fig. 3. Distribution of publication years

TABLE VI
PRIMARY STUDIES INCLUDED, WITH THEIR TYPES AND NAMES OF APPROACHES.

ID	Reference	Tool	Plugin	Name
S1	[11]			
S2	[13]			
S3	[15]	x		
S4	[5]	x		COFAQ
S5	[18]	x		AdDoc
S6	[20]		x	eMoose
S7	[22]	x		Apatite
S8	[24]			
S9	[26]		x	Codesnippets
S10	[28]			
S11	[30]			
S12	[32]	x		
S13	[34]	x		
S14	[36]	x		FUDA
S15	[38]			
S16	[40]	x		CommentWeaver
S17	[42]			
S18	[44]	x		
S19	[12]			
S20	[14]			
S21	[16]	x		ProperDoc
S22	[17]	x		Haddock
S23	[19]			
S24	[21]	x		APIMiner
S25	[23]	x		ExPort
S26	[25]			
S27	[27]			
S28	[29]			
S29	[31]			
S30	[33]	x		Jadeite
S31	[35]	x		Baker
S32	[37]	x		SISE
S33	[39]			
S34	[41]		x	CoDocent
S35	[43]	x		DocRef
S36	[45]	x		Doc2Spec

B. Contributions to API documentation (RQ2)

Table VII shows what each of the approaches generate as output. The classification of outputs were generated in a two step procedure. In the first step, the exact output from each of the approaches was documented, resulting in a quite diverse table of outputs. Therefore, in the second step, these outputs were grouped together on a higher level of abstraction, producing the final table shown below. To minimize researcher bias and human error, the final table was double checked by two researchers and any disagreements were resolved through discussions.

The majority of the approaches contribute by generating new documentation in some format. From Table VII, it is evident that code examples and templates along with natural language (NL) documentation are the most popular targets for documentation generation. These targets for documentation generation are understandable, since lacking code examples or erroneous, inadequate or lacking descriptions are common reasons for many API documents being difficult to understand and use (see e.g. [16]). However, specifications and usage rules are almost as an appealing target, as is visual and navigational enhancement. These four targets for tool support for API documentation suggest that API documentation easily becomes very complex and hard to understand, and additionally it may be difficult to find the relevant documentation for a given problem because of the size of the documentation. Five studies address recommendations and identifications, and these studies mostly provide information on what needs to be altered in the API documentation, but also recommend which classes to use for a specific problem. As a consequence, most of these studies assist in both the maintenance and generation of the documentation, and in the accuracy and correctness of it. While only five studies address recommendations and identifications, of which three contribute to the accuracy and correctness of the API documentation, there are many other approaches that also contribute to the correctness of the documentation (see Table IX).

C. Sources for API documentation (RQ3)

In order to get the desired output from an API documentation approach, some specific sources need to be available. With this in mind, Table VIII lists the sources required by the approaches for assisting in API documentation generation. The classification of sources were generated in a similar two step procedure as described in Section III-B. Clearly, source code, code comments, and code examples are by far the most used sources for documentation generation. This can easily be understood since the source code is the basis on which the API documentation should ultimately be generated upon in any API. API usage in existing software is the second most used source, and in most approaches, these calls are used as basis for generating either examples or usage rules. An interesting thing to note is that eight of the documentation approaches use API documentation and tutorials as input, i.e., from existing documentation new documentation is generated. Many of these approaches are based on natural language processing for automatically analyzing the written descriptions. Quite many approaches also rely on online information, predominantly on forum discussions found on Stack Overflow. These forum discussions are mainly used for collecting information on API usages and then creating, e.g. examples based on these, or explanatory descriptions for how to use the API in certain situations.

D. Properties of Documentation Approaches (RQ4)

The different approaches for API documentation generation tackle the documentation problem from different perspectives. Consequently, they contribute to API documentation in different ways, and therefore assist both API users and API documentation writers in different ways. Table IX lists the properties that the corresponding API documentation approaches have. Again, the classification of properties were generated in a similar two step procedure as described in Section III-B. However, we point out that not all of the properties listed in Table IX were explicitly mentioned in the primary studies. The authors of this paper have in some cases taken the liberty of deducing the properties on their own, based on the discussions in the studies.

TABLE VII
OUTPUT FROM API DOCUMENTATION APPROACHES

Output	Count	Primary Studies	Description
Examples & Templates	11	S9, S14, S15, S17, S18, S21, S23, S24, S25, S27, S30	Produces examples of how to use the API or basic structures for using the API
NL Documentation	10	S2, S3, S4, S11, S16, S19, S22, S28, S29, S32	Produces or augments the natural language description in the API documentation
Specifications & Rules	9	S1, S2, S10, S12, S13, S20, S26, S33, S36	Produces specifications (e.g. formal specifications) and usage rules
Visual & Navigational Enhancement	9	S6, S7, S8, S25, S28, S30, S31, S32, S34	Visual representation of API or simplified browsing in the documentation
Recommendations & Identifications	5	S5, S7, S12, S30, S35	Recommendations of different kinds (e.g. documentation in need of attention, which class to use), identifications of e.g. incorrect descriptions

TABLE VIII
SOURCES FOR API DOCUMENTATION APPROACHES

Source	Count	Primary Studies	Description
Source Code & Examples	18	S1, S2, S5, S7, S8, S9, S11, S12, S13, S16, S19, S22, S24, S28, S31, S32, S34, S35	Source code, code comments and annotations, example code
API Usage & API calls	9	S1, S3, S14, S17, S20, S23, S25, S30, S33	API usage scenarios, traces of method calls in existing software
API Documentation & Tutorials	8	S5, S6, S10, S23, S26, S27, S35, S36	Natural language description and API tutorials
Online Information	7	S4, S11, S18, S29, S31, S32, S34	Websites such as Stack Overflow
Manual Input	5	S9, S12, S14, S15, S21	Requires manual input of API user
Databases & Search Engines	4	S4, S7, S21, S34	Results from search engines, data found in databases

TABLE IX
PROPERTIES OF API DOCUMENTATION APPROACHES

Properties	Count	Primary Studies	Description
Understandability & Learnability	17	S2, S6, S9, S10, S11, S12, S15, S17, S18, S21, S24, S25, S30, S31, S32, S33, S34	Improves the users understanding of the API and helps the user learn the API
Maintenance & Generation	12	S4, S5, S12, S16, S17, S19, S22, S28, S29, S31, S35, S36	Supports API documentation maintenance, or generates new documentation
Accuracy & Correctness	11	S1, S2, S3, S5, S12, S13, S15, S20, S22, S26, S35	Tries to verify that the documentation is without errors, and that descriptions are accurate
Accessibility & Structure	9	S4, S7, S8, S23, S27, S29, S30, S31, S34	Assists in finding relevant information, or improves the overall API documentation structure
Productivity	5	S14, S18, S24, S30, S34	Enhances API users productivity
Usability & Reusability	5	S1, S2, S6, S26, S29	Supports usability of API or reusability of API during development

As can be seen from Table IX, the most addressed property of the approaches is to contribute to the understandability and learnability of documentation, which is understandable, since knowing an API will help an API user notably in achieving the desired development goals. The second most addressed property is the maintenance and generation of API documentation, which also is understandable since both maintenance and generation can be time consuming and error prone, specifically in the context of large APIs. Many approaches also contribute to the accuracy and correctness of API documention, in many cases with formal specifications, and to API accessibilty and structure, making information finding easier in the documentation. Finding relevant information in API documentation is repeatedly mentioned in studies on API documentation to be a current problem. These observations suggest that these properties are the most lacking in current API documentation.

We note that the *Productivity* class is a highly undescriptive one, since it is largely a combination of the other classes listed in the table. However, improved productivity as a result of improved API documentation is likely one of the most desirable goals. We have therefore decided to include this class of properties, but it should be noted that most of the other classes listed in Table IX should indirectly contribute to improved productivity as well.

E. Evaluation Methods (RQ5)

The evaluation of API documentation approaches is probably as important as the approach itself. Without an evaluation, there is no evidence on whether the approach actually contributes to the API documentation as intended, nor if the result of the approach makes any difference. Somewhat surprisingly, not all of the primary studies offered an evaluation of the approach presented. It is however possible that evaluations of the approaches have been done in studies published afterwards, but which are not included in this SMS.

Figure 4 shows the distribution of the chosen methods for evaluation for the approaches. Exercise of the proposed approach, where the authors evaluated the output from the approach, was the dominant evaluation method, but other forms of empirical evaluation methods were also used in

many studies. The class *Analytical* refers to statistical analysis of the approach, the class *Review* refers to the approaches being reviewed by experts, while the class *Comparison* refers to the authors comparing their approach with other similar approaches. While seven of the approaches were not evaluated at all, equally many approaches were evaluated with more than one evaluation method. This is the reason to why the total count of evaluations is larger than the number of primary studies in this paper.

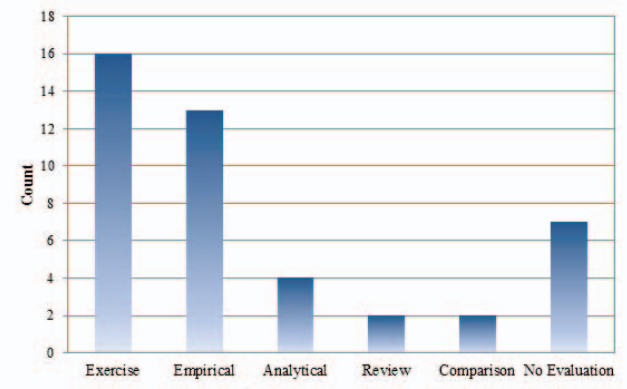

Fig. 4. Evaluation methods for the approaches

IV. Threats to Validity

A threat to the validity of this study is that the quality assessment and data extraction was done by one author only. This means that the results may be slightly biased, and that some facts reported in the primary studies may have been misunderstood, or missed. Still, the results presented in Tables VII, VIII, and IX have been double checked by two researchers, which partially reduces this threat to the validity of the results.

Another threat to the validity stems from the difficulty in searching for original papers on API documentation. As mentioned in Section III, the search resulted in a large number of papers, primarily because the search terms included words such as ”document” and ”library”. Although these words are relevant in the context of API documentation, they are even more relevant in the context of physical libraries, and performing the title and abstract screening on this amount of papers may have resulted in some papers being filtered out accidentally, even though the screening was performed by two researchers.

Finally, this study is based on publications within the scientific community. There may therefore exist API documentation approaches that are not included in this study if they have not been presented in scientific publications.

V. Conclusions

In this paper, we presented a systematic mapping study on approaches for Application Programming Interface (API) documentation generation. The systematic mapping study itself was also presented for replicability. Since the beginning of the millennium, there have been many contributions to this topic, and many approaches have been developed. We identified 36 studies on this topic, which we analyzed based on the research questions detailed in Section II.

Out of the identified approahces, many contribute to API documentation in the areas of natural language documentation and code examples. This suggests that current API documentation is in general lacking in these, or are at least in need for tool support. Many of the approaches were designed to ease developers understanding and learning of API's, but also to contribute to the maintenance and generation of API documentation. Most of the approaches were automatic, simplifying the API documentation generation notably, under the assumption that relevant sources for the generation are availabe. Empirical evaluation and exercise of the proposed approach were the dominiant evaluation methods in the primary studies, but some papers also provided extensive evaluations using several different evaluation methods.

With this study, we hope to give interested readers an overview of what API documentation approaches have been published and of their properties. However, we note that some API documentation approaches may not be included in this study, since not all such approaches are published within the scientific community. Instead, they may have been released as ready products, but since this study was based on published papers, such approaches are not included in this paper.

Acknowledgements This work has been partially supported by the Dimecc Need for Speed program and funded by Tekes, the Finnish Funding Agency for Technology and Innovation. This work has received funding from the Electronic Component Systems for European Leadership Joint Undertaking under grant agreement No 737494. This Joint Undertaking receives support from the European Union's Horizon 2020 research and innovation programme and Sweden, France, Spain, Italy, Finland, Czech Republic.

References

[1] M. Henning, “Api design matters,” *ACM Queue*, vol. 5, pp. 24–36, May 2007.

[2] M. F. Zibran, “What makes APIs difficult to use?,” *International Journal of Computer Science and Network Security*, no. 4, pp. 255–261.

[3] J. Andersson, S. Larsson, M. Ericsson, and A. Wingkvist, “A study of demand-driven documentation in two open source projects,” in *2015 48th Hawaii International Conference on System Sciences*, pp. 5271–5279, Jan 2015.

[4] Y. Acar, M. Backes, S. Fahl, D. Kim, M. L. Mazurek, and C. Stransky, “You get where you're looking for: The impact of information sources on code security,” in *2016 IEEE Symposium on Security and Privacy (SP)*, pp. 289–305, May 2016.

[5] C. Chen and K. Zhang, “Who asked what: Integrating crowdsourced faqs into api documentation,” in *Companion Proceedings of the 36th International Conference on Software Engineering*, ICSE Companion 2014, (New York, NY, USA), pp. 456–459, ACM, 2014.

[6] S. C. B. de Souza, N. Anquetil, and K. M. de Oliveira, “A study of the documentation essential to software maintenance,” in *Proceedings of the 23rd Annual International Conference on Design of Communication: Documenting &Amp; Designing for Pervasive Information*, SIGDOC '05, pp. 68–75, 2005.

[7] B. Kitchenham and S. Charters, “Guidelines for performing Systematic Literature Reviews in Software Engineering (version 2.3),” Tech. Rep. EBSE-2007-01, Keele University and University of Durham, 2007.

[8] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering*. Springer-Verlag Berlin Heidelberg, 1 ed., 2012.

[9] F. Mateen, A. Tergas, N. Bhayani, and B. Kamdar, "Titles versus titles and abstracts for initial screening of articles for systematic reviews," *Clinical Epidemiology*, vol. 5, pp. 89–95, 2013.

[10] M. Usman, E. Mendes, F. Weidt, and R. Britto, "Effort estimation in agile software development: A systematic literature review," in *Proceedings of the 10th International Conference on Predictive Models in Software Engineering*, PROMISE '14, (New York, NY, USA), pp. 82–91, ACM, 2014.

[11] M. Acharya, T. Xie, J. Pei, and J. Xu, "Mining api patterns as partial orders from source code: From usage scenarios to specifications," in *Proceedings of the the 6th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering*, ESEC-FSE '07, (New York, NY, USA), pp. 25–34, ACM, 2007.

[12] D. M. Leslie, "Using javadoc and xml to produce api reference documentation," in *Proceedings of the 20th Annual International Conference on Computer Documentation*, SIGDOC '02, (New York, NY, USA), pp. 104–109, ACM, 2002.

[13] M. Bruch, M. Mezini, and M. Monperrus, "Mining subclassing directives to improve framework reuse," in *2010 7th IEEE Working Conference on Mining Software Repositories (MSR 2010)*, pp. 141–150, May 2010.

[14] D. Lo, G. Ramalingam, V.-P. Ranganath, and K. Vaswani, "Mining quantified temporal rules: Formalism, algorithms, and evaluation," *Science of Computer Programming*, vol. 77, no. 6, pp. 743 – 759, 2012. (1) Coordination 2009 (2) {WCRE} 2009.

[15] R. P. Buse and W. R. Weimer, "Automatic documentation inference for exceptions," in *Proceedings of the 2008 International Symposium on Software Testing and Analysis*, ISSTA '08, (New York, NY, USA), pp. 273–282, ACM, 2008.

[16] L. W. Mar, Y. C. Wu, and H. C. Jiau, "Recommending proper api code examples for documentation purpose," in *2011 18th Asia-Pacific Software Engineering Conference*, pp. 331–338, Dec 2011.

[17] S. Marlow, "Haddock, a haskell documentation tool," in *Proceedings of the 2002 ACM SIGPLAN Workshop on Haskell*, Haskell '02, (New York, NY, USA), pp. 78–89, ACM, 2002.

[18] B. Dagenais and M. P. Robillard, "Using traceability links to recommend adaptive changes for documentation evolution," *IEEE Transactions on Software Engineering*, vol. 40, pp. 1126–1146, Nov 2014.

[19] C. McMillan, D. Poshyvanyk, and M. Grechanik, "Recommending source code examples via api call usages and documentation," in *Proceedings of the 2Nd International Workshop on Recommendation Systems for Software Engineering*, RSSE '10, (New York, NY, USA), pp. 21–25, ACM, 2010.

[20] U. Dekel and J. D. Herbsleb, "Improving api documentation usability with knowledge pushing," in *2009 IEEE 31st International Conference on Software Engineering*, pp. 320–330, May 2009.

[21] J. E. Montandon, H. Borges, D. Felix, and M. T. Valente, "Documenting apis with examples: Lessons learned with the apiminer platform," in *2013 20th Working Conference on Reverse Engineering (WCRE)*, pp. 401–408, Oct 2013.

[22] D. S. Eisenberg, J. Stylos, A. Faulring, and B. A. Myers, "Using association metrics to help users navigate api documentation," in *2010 IEEE Symposium on Visual Languages and Human-Centric Computing*, pp. 23–30, Sept 2010.

[23] E. Moritz, M. Linares-Vsquez, D. Poshyvanyk, M. Grechanik, C. McMillan, and M. Gethers, "Export: Detecting and visualizing api usages in large source code repositories," in *2013 28th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pp. 646–651, Nov 2013.

[24] H. Eriksson, E. Berglund, and P. Nevalainen, "Using knowledge engineering support for a java documentation viewer," in *Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering*, SEKE '02, (New York, NY, USA), pp. 57–64, ACM, 2002.

[25] R. Pandita, X. Xiao, H. Zhong, T. Xie, S. Oney, and A. Paradkar, "Inferring method specifications from natural language api descriptions," in *2012 34th International Conference on Software Engineering (ICSE)*, pp. 815–825, June 2012.

[26] A. Forward, T. Lethbridge, and D. Deugo, "Codesnippets plug-in to eclipse: Introducing web 2.0 tagging to improve software developer recall," in *5th ACIS International Conference on Software Engineering Research, Management Applications (SERA 2007)*, pp. 451–460, Aug 2007.

[27] G. Petrosyan, M. P. Robillard, and R. D. Mori, "Discovering information explaining api types using text classification," in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, vol. 1, pp. 869–879, May 2015.

[28] C. Gao and J. Wei, "Generating open api usage rule from error descriptions," in *2013 IEEE Seventh International Symposium on Service-Oriented System Engineering*, pp. 245–253, March 2013.

[29] R. Pierce and S. Tilley, "Automatically connecting documentation to code with rose," in *Proceedings of the 20th Annual International Conference on Computer Documentation*, SIGDOC '02, (New York, NY, USA), pp. 157–163, ACM, 2002.

[30] L. Guerrouj, D. Bourque, and P. C. Rigby, "Leveraging informal documentation to summarize classes and methods in context," in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, vol. 2, pp. 639–642, May 2015.

[31] L. B. L. d. Souza, E. C. Campos, and M. d. A. Maia, "On the extraction of cookbooks for apis from the crowd knowledge," in *2014 Brazilian Symposium on Software Engineering*, pp. 21–30, Sept 2014.

[32] J. Henkel, C. Reichenbach, and A. Diwan, "Developing and debugging algebraic specifications for java classes," *ACM Trans. Softw. Eng. Methodol.*, vol. 17, pp. 14:1–14:37, June 2008.

[33] J. Stylos, A. Faulring, Z. Yang, and B. A. Myers, "Improving api documentation using api usage information," in *2009 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pp. 119–126, Sept 2009.

[34] J. Henkel, C. Reichenbach, and A. Diwan, "Discovering documentation for java container classes," *IEEE Transactions on Software Engineering*, vol. 33, pp. 526–543, Aug 2007.

[35] S. Subramanian, L. Inozemtseva, and R. Holmes, "Live api documentation," in *Proceedings of the 36th International Conference on Software Engineering*, ICSE 2014, (New York, NY, USA), pp. 643–652, ACM, 2014.

[36] A. Heydarnoori, K. Czarnecki, W. Binder, and T. T. Bartolomei, "Two studies of framework-usage templates extracted from dynamic traces," *IEEE Transactions on Software Engineering*, vol. 38, pp. 1464–1487, Nov 2012.

[37] C. Treude and M. P. Robillard, "Augmenting api documentation with insights from stack overflow," in *Proceedings of the 38th International Conference on Software Engineering*, ICSE '16, (New York, NY, USA), pp. 392–403, ACM, 2016.

[38] D. Hoffman and P. Strooper, "API documentation with executable examples," *Journal of Systems and Software*, vol. 66, no. 2, pp. 143 – 156, 2003.

[39] C. C. Williams and J. K. Hollingsworth, "Recovering system specific rules from software repositories," *SIGSOFT Softw. Eng. Notes*, vol. 30, pp. 1–5, May 2005.

[40] M. Horie and S. Chiba, "Tool support for crosscutting concerns of api documentation," in *Proceedings of the 9th International Conference on Aspect-Oriented Software Development*, AOSD '10, (New York, NY, USA), pp. 97–108, ACM, 2010.

[41] Y. C. Wu, L. W. Mar, and H. C. Jiau, "Codocent: Support api usage with code example and api documentation," in *2010 Fifth International Conference on Software Engineering Advances*, pp. 135–140, Aug 2010.

[42] J. Jiang, J. Koskinen, A. Ruokonen, and T. Systa, "Constructing usage scenarios for api redocumentation," in *15th IEEE International Conference on Program Comprehension (ICPC '07)*, pp. 259–264, June 2007.

[43] H. Zhong and Z. Su, "Detecting api documentation errors," *SIGPLAN Not.*, vol. 48, pp. 803–816, Oct. 2013.

[44] J. Kim, S. Lee, S.-W. Hwang, and S. Kim, "Enriching documents with examples: A corpus mining approach," *ACM Trans. Inf. Syst.*, vol. 31, pp. 1:1–1:27, Jan. 2013.

[45] H. Zhong, L. Zhang, T. Xie, and H. Mei, "Inferring resource specifications from natural language api documentation," in *Proceedings of the 2009 IEEE/ACM International Conference on Automated Software Engineering*, ASE '09, (Washington, DC, USA), pp. 307–318, IEEE Computer Society, 2009.